

A3
21. (Amended) The method of claim 1, wherein the prefetch instruction includes a field that identifies an instruction to prefetch from memory into the high speed buffer.

22. (Amended) The method of claim 11, wherein the prefetch instruction includes a field that identifies data to prefetch from memory into the high speed buffer, wherein the data is operated on by at least one instruction in the instruction stream.

REMARKS/ARGUMENTS

Status of Application.

Claims 1, 3-12, and 14-22 remain in the application. Claims 2 and 13 have been cancelled. Claims 1, 5, 7, 12, 21, and 22 have been amended.

Information Disclosure Statement.

The Office Action states that the information disclosure statement filed August 3, 2000 failed to include a copy of reference AC. Applicants believe that reference AC listed as U.S. Patent 5,843,934 was included by error and that the intended reference was U.S. Patent No. 5,854,934: Optimizing Compiler Having Data Cache Prefetch Spreading. A supplemental Information Disclosure Statement is being filed enclosing a copy of this reference.

Declaration.

The Office Action found that the Declaration did not comply with 37CFR §§ 602.01 and 602.02 for failing to identify the mailing or post office address of each inventor. Applicants submit that the addresses set forth in the declaration are the same as the Post Office Addresses.

Drawings.

The Office Action objected to the drawings for informalities (incorrect upper and

left margins). The undersigned discussed the drawing informalities with the Examiner on September 23, 2002 and agreed to provide formal drawings promptly after this response. The drawing informalities consist of margin errors.

Specification.

The Office Action objected to the specification for certain informalities. Specifically, the Office Action states that the Summary of the Invention and Cross Reference to Related Inventions are missing. Applicants are therefore amending the application to include both a Summary and Cross Reference.

Double Patenting.

The Office Action provisionally rejected claims 1, 6, and 12 under 35 USC § 101 as claiming the same invention as that of claims 1, 14, and 18 of the co-pending application 09/459,739. Applicants submit that the rejection has been overcome by the amendment of claims 1 and 12. Applicants traverse the rejection on grounds that the claims of the 09/459,739 application differ patentably from those being rejected herein. Moreover, claim 1 has been amended to change the step of generating first path data to deriving first path data from a compiler and claims 1, 14, and 18 of the co-pending application 09/459,739 do not teach or suggest that limitation.

Claims 10, 11, 21, and 22 were rejected under the doctrine of obviousness-type double patenting as being unpatentable over claims 16, 17, 33, and 34 of copending Application No. 09/459,739. Applicant is submitting a terminal disclaimer with the 09/459,739 application and that overcomes this rejection.

Claim Rejections – 35 U.S.C. § 112.

The Office Action rejected claims 21 and 22 under 35 U.S.C. § 112, second paragraph on grounds that those claims are apparatus claims that depend on method

claims. Accordingly, the preamble of both claims has been rewritten to relate to methods.

Claim Rejections – 35 U.S.C. § 102.

Claims 1, 6, and 12 were rejected under 35 U.S.C. § 102(f) on grounds that the applicant did not invent the claimed subject matter. Applicants respectfully submit that claims 1, 6, and 12 define an invention that is patentably distinct from the inventions for reasons discussed above in the response to the double-patenting rejections.

Claims 1-5, 10, 12-16, and 21 were rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. patent number 5,742,804 to Yeh (hereafter, Yeh). Applicants respectfully submit that claims 1-5, 10, 12-16, and 21, as amended, are not anticipated by Yeh for the reasons discussed below.

Yeh relates to a method and system for reducing the instruction fetch penalty in the execution of a programmed sequence and includes inserting a branch predict instruction that specifies a branch taken or not taken. Branch prediction instructions are inserted into a program by a compiler to aid in the prediction of the branch and prefetch instructions into a cache before their use by a processor. Included within the format of the branch prediction instruction are:

- an offset from the branch prediction instruction to the branch;
- a taken/not-taken bit that denotes a static prediction made by the compiler indicating if the designated branch is predicted as taken or not-taken.;
- a size field indicating the amount of information that should be prefetched;
- a target field indicating the target address of the branch; and
- a trace vector indicating the path from where the branch prediction instruction is located to the target branch.

Figure 1 of Yeh illustrates a branch prediction instruction inserted into a program. The figure shows a basic block tree structure of a program. At the end of each basic block is a branch indicating the program flow to subsequent basic blocks. The branch

positions are numbered 0 through 8. A Predict-branch instruction is inserted at the top of the tree to provide hint information about branch #6 and target information regarding a prefetch address and amount of information to prefetch. The trace field in the Predict-branch instruction indicates the branch-path from the branch prediction instruction to the prefetch target.

Prefetches can be canceled by predicting an executed path of the programmed sequence of instruction from the predict-branch instruction to the targeted branch instruction and then comparing the trace vector (obtained from the Predict-branch instruction) to the predicted execution path (obtained from the branch predictor at run-time).

The claimed invention is significantly different from the mechanism described by Yeh.

First, claim 1 was amended to state that the method includes a step of prefetching instructions and data when a stated condition occurs. The mechanism described by Yeh can only prefetch instructions and cannot prefetch data. The prediction-branch instruction described by Yeh is tied to the branch and the prefetch address is tied to the target of the branch. Yeh has no mechanism to generate a prefetch address corresponding to a data or operand address. Additionally, the title of the Yeh patent further provides more evidence that this mechanism can only prefetch instructions: "Instruction Prefetch Mechanism Utilizing a Branch Predict Instruction".

Support for the above amendment is found at page 10, line 20: "During compilation, prior to the compiler's first pass of instruction scheduling, a touch instruction is inserted immediately preceding every memory reference (load or store or program segment), and the touch register results are included as an operand to the subject memory reference." Clearly the claimed mechanism is prefetching instructions and data. The touch [prefetch] instruction is not tied to a branch, only to the operand address or program segment that uses the perfected information. During successive passes of

instruction scheduling, the compiler attempts to 'move up' each touch instruction to cover all or part of the expected memory latency to their dependent memory references.

Second, claim 1 requires the following step: "deriving first path data from a compiler performing static compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction." The Office Action cites col. 4, lines 8-23 of Yeh for this limitation. That section relates to the insertion of a branch predict instruction and does not relate to path data representing a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction.

Third, the claimed invention also uses a new branch prediction mechanism where the branch path history is obtained from branch information generated by the branch encountered prior to the touch instruction. Claim 1 has been amended to include the step of: "obtaining a branch path history from branch information generated by a branch encountered prior to the prefetch instruction." Yeh neither teaches nor suggests this step.

For example, Figure 2 of Yeh shows the program tree used in describing Yeh's prefetching mechanism plus the two previous basic blocks encountered prior to the branch-predict instruction, denoted -1 and -2. In the claimed mechanism the branch history mask, used to compare to the branch mask in the prefetch instruction to determine if the prefetch instruction should be executed, is generated even before a prefetch instruction is encountered. Using the prefetch example of Figure 1 of Yeh, the branch history mask for the Predict-branch #6 in block 0 is generated when basic blocks -2 or -1 of Figure 2 are entered. In the Yeh patent the predicted path history is developed by the branch prediction mechanism after the prefetch instruction is encountered.

Thus claim 1 requires deriving the branch path from the prefetch instruction to the instructions that use the prefetched information earlier in the processor's pipeline than the Yeh patent. There are several advantages in developing the branch path (branch-history-mask) even before the touch (prefetching) instruction is encountered. One is that no time elapses between decoding the prefetch instruction and canceling the prefetch instruction if

the branch mask in the prefetch instruction does not match the branch history mask. Clearly, this is a result of having the branch history mask developed even before the prefetch instruction is decoded. The Yeh patent neither teaches developing the branch history mask before the predict-branch is encountered nor does it describe a mechanism that can do this. Another advantage is that the claimed branch prediction mechanism is available for future inquiries to predict the action of future branches. It is not over engaged in predicting future branches as well as predicting a branch path to the target branch as described in the patent by Yeh.

Claim 3 requires that the second path data is derived from information characterizing dynamic execution of the sequence of instructions by the at least one processor unit. The Office Action cites Yeh, col. 6, lines 28-33. Again, the second path is the predicted path. The Office Action contends that the execution path corresponds to the second path. Applicant respectfully traverses that finding. Moreover, claim 3 (as well as claims 4, 5, 10, 12-16, and 21) is not anticipated by Yeh for the same reasons that claim 1 is not. Claim 4 is also not anticipated by Yeh for the same reasons that claim 1 is not.

Claim Rejections – 35 U.S.C. § 103.

Claims 6-9, 11, 17-20, and 22 were rejected under 35 U.S.C. § 103(a) as unpatentable over Yeh. Applicants respectfully contend that claims 6-9, and 11 are all dependent on claim 1 and hence would not have been obvious for the same reasons that claim 1 would not have been obvious in view of Yeh. Claims 17-20 and 22 are dependent on claim 12. Claim 12 has been amended to include limitations equivalent to those of claim 1 and hence it and its dependent claims are patentable for the same reasons that claim 1 is.

Attached hereto is a marked-up version of the changes made to the specifications and claims by the current amendment. The attached page is captioned “**Version with markings to show changes made.**”

Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,

By: Michael J. Buchenhorner

Michael J. Buchenhorner

Registration No. 33,162

Telephone No.: (305) 529-0221



VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the specification:

At page 1, following the title of the invention, please delete the phrase "Background of the Invention" and insert the following paragraphs:

-- CROSS-REFERENCE TO RELATED APPLICATIONS.

This application relates to subject matter that is similar to that of the copending application having serial number 09/459,739 which was also filed on December 10, 1999 and is assigned to the same entity.

STATEMENT REGARDING FEDERALLY-SPONSORED RESEARCH OR DEVELOPMENT.

Not Applicable.

INCORPORATION BY REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC.

Not Applicable.

BACKGROUND OF THE INVENTION. --

At page 6, line 20, please insert the following paragraph:

-- SUMMARY OF THE INVENTION.

In a system including a high speed buffer logically placed between memory and at least one processor unit, a method for executing an instruction stream stored in the memory, wherein the instruction stream comprises a sequence of instructions including at least one prefetch instruction that prefetches information from the memory into the high speed

buffer, the method comprises the following steps: deriving first path data from a compiler performing static compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction; generating second path data, wherein the second path data represents a predicted second path of execution; determining whether the first path falls within the predicted second path; and prefetching instructions and data when the first path falls within the predicted second path. --

In the claims:

Claims 1, 5, 7, 12, 21, and 22 have been amended as follows:

1. (Amended) In a system including a high speed buffer logically placed between memory and at least one processor unit, a method for executing an instruction stream stored in the memory, wherein the instruction stream comprises a sequence of instructions including at least one prefetch instruction that prefetches information from the memory into the high speed buffer, the method comprising the steps of:

[generating] deriving first path data from a compiler by analyzing control flow information during compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction;

obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction;

generating second path data, wherein the second path data represents a predicted second path of execution;

determining whether the first path is consistent with the predicted second path;
and

[conditionally executing the prefetch instruction] prefetching instructions and data [based upon a comparison operation that compares the first path data to the second path data to determine if the first path falls within the predicted second path] when the first path is consistent with the predicted second path.

2. (Cancelled) The method of claim 1, wherein the first path data is derived from a compiler performing static compilation.

5. (Amended) The method of claim 1, further comprising the step of:
upon determining that the first [paths] path does not fall within the predicted second path, omitting the prefetch instruction from the instruction stream executed by the at least one processor unit.

7. (Amended) The method of claim 6, wherein the first path data comprises a mask that represents a path of execution from the prefetch instruction to the instruction that uses the information prefetched by the prefetch instruction.

12. (Amended) In a system including a memory storing an instruction stream comprising a sequence of instructions including at least one prefetch instruction, a processor unit for executing the sequence of instructions, and a high speed buffer logically placed between the memory and the at least one processor unit, an apparatus for conditionally executing the prefetch instruction comprising:

decode logic for [generating] deriving first path data from a compiler performing static compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction;

logic for obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction;

path prediction logic for generating second path data, wherein the second path data represents a predicted second path of execution;

• compare logic for determining whether the first path is consistent with the predicted second path; and

• execution logic for conditionally [executing the prefetch instruction] prefetching instructions and data [based upon a comparison operation that compares the first path data to the second path data to determine if the first path falls within the predicted second path] when the first path is consistent with the predicted second path.

21. (Amended) The [apparatus] method of claim 1, wherein the prefetch instruction includes a field that identifies an instruction to prefetch from memory into the high speed buffer.

22. (Amended) The [apparatus] method of claim 11, wherein the prefetch instruction includes a field that identifies data to prefetch from memory into the high speed buffer, wherein the data is operated on by at least one instruction in the instruction stream.